Spring REST

RestController

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Rest isNetwork Architectural style
- Resources are defined and addressed
- Operations mapped to URLs
- RestController provides Restful service in spring MVC

Naming Resources

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- REST uses URI to identify resources

  http://localhost/books/
  http://localhost/books/ISBN-0011
  http://localhost/books/ISBN-0011/authors

  http://localhost/classes
  http://localhost/classes/cs2650
  http://localhost/classes/cs2650/students

- As you traverse the path from more generic to more specific, you are navigating the data

Verbs

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Represent the actions to be performed on resources

- HTTP GET
- HTTP POST
- HTTP PUT
- HTTP DELETE

HTTP GET

2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207

- How clients ask for the information they seek.
- Issuing a GET request transfers the data from the server to the client in some representation

- GET http://localhost/books
  - Retrieve all books
- GET http://localhost/books/ISBN-0011021
  - Retrieve book identified with ISBN-0011021
- GET http://localhost/books/ISBN-0011021/authors
  - Retrieve authors for book identified with ISBN-0011021

HTTP PUT, POST, DELETE

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- HTTP POST creates a resource
- HTTP PUT updates a resource
- HTTP DELETE removes a resource

- POST http://localhost/books/
  - Content: {title, authors[], …}
  - Creates a new book with given properties

- PUT http://localhost/books/isbn-111
  - Content: {isbn, title, authors[], …}
  - Updates book identified by isbn-111 with submitted properties

- DELETE http://localhost/books/ISBN-0011
  - Delete book identified by ISBN-0011

Typical REST URLs

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

REST Endpoint	HTTP Method	Description
/customers	GET	Returns the list of customers
/customers/{id}	GET	Returns customer detail for given customer {id}
/customers	POST	Creates new customer from the post data
/customers	PUT	Replace the details for given customer
/customers/{id}	DELETE	Delete the customer for given customer {id}

Representations

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- XML

```
<COURSE>
    <ID>CS2650</ID>
    <NAME>Distributed Multimedia Software</NAME>
</COURSE>
```

- JSON

```
Course :
    {“id”: “CS2650”,
    “name”: “Distributed Multimedia Sofware”}
```

@RequestMapping

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- used on classes and such classes are referred to as root resource classes

- @RequestMapping("calc") sets the path to the base URI + /calc. The base URI consists of the host, port and any context.

- HTTP requests can also be mapped through this annotation
- Can also specify type odf data consumed and type of data produced
- Example:

```
@RequestMapping(path = "/",
consumes = MediaType.APPLICATION_JSON_VALUE,
produces = MediaType.APPLICATION_JSON_VALUE,
method = RequestMethod.GET)
```

@RequestBody

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- *@RequestBody* annotation maps the *HttpRequest* body to a domain object, enabling automatic deserialization of the inbound *HttpRequest* body onto a Java object
- Example:

```
@RequestMapping("/request")
public ResponseEntity postController(  @RequestBody Emp x) {
    //   ……
  }
```

ResponseEntity

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- ResponseEntity embeds the response object along with HTTP info
- Useful when the method returns different types of data and also to send HTTP response codes

REST Client : RestTemplate

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Accessing a REST service inside a Spring application revolves the use of the Spring **RestTemplate** class.
- The RestTemplate class is designed on the same principles as the many other Spring Template classes likeJdbcTemplate, & JmsTemplate providing a simplified approach with default behaviors for performing complex tasks
- Example:

```
RestTemplate restTemplate = new RestTemplate();
String result = restTemplate.getForObject(uri, String.class);
```

- RestTemplate class has methods
  - headForHeaders()
  - getForObject()
  - postForObject()
  - put()
  - delete()
  - ……. etc

RestTemplate Examples

2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207

```
HttpHeaders headers = new HttpHeaders();
headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));

HttpEntity<String> entity = new HttpEntity<String>("parameters", headers);

ResponseEntity<String> result = restTemplate.exchange(uri, HttpMethod.GET,
entity, String.class);
```

```
String uri = "http://localhost:8080/springrestexample/employees/{id}";

Map<String, String> params = new HashMap<String, String>();
params.put("id", "1");

RestTemplate restTemplate = new RestTemplate();
Employee result = restTemplate.getForObject(uri, Employee.class, params);
```